TECHNICAL REPORT DELLA TASK DI RECRUITING PER SW-DRIVERLESS IN EAGLE

In questo report verranno discusse, per ogni livello della task di recruiting:

- Strategia risolutiva;
- Difficoltà riscontrate e come sono state superate;
- Possibili miglioramenti, ove possibile;

I punti successivi sono stati implementati lavorando direttamente sulle immagini scattate sulla macchina che percorre un circuito delimitato da due coni rossi a destra e sinistra seguiti da due file di coni blu e gialli rispettivamente. I coni rossi e blu hanno una o più strisce bianche al centro, mentre i coni gialli hanno soltanto una striscia nera al centro.

LIVELLO 1: LOAD AND DISPLAY CAPTURED DATA

La richiesta è stata soddisfatta istanziando un oggetto di tipo *cv::Mat* con percorso assoluto del frame. Se l'immagine viene trovata regolarmente, essa viene stampata a video tramite la funzione *imshow* della libreria OpenCV. In questo punto non sono state riscontrate difficoltà. L'implementazione è già ottimale perché è accessibile completamente in OpenCV.

LIVELLO 2: CONE DETECTION

Per trovare i coni all'interno dell'immagine è stata implementata una pipeline di image processing utilizzando API presenti in OpenCV. Di seguito vengono riportate le operazioni in ordine di applicazione:

1) Cambiamento dello spettro dei colori da BGR a HSV al fine di rilevare meglio i colori;
2) Filtro sui colori primario e secondario di ogni cono (1);

    a. Poiché nero e bianco sono colori “secondari” in un cono, poiché rappresentano linee intermedie e quindi non permettono di classificare l’oggetto per il suo colore effettivo, sono i primi colori a essere filtrati;
    b. Dopo aver filtrato rosso o blu, alla maschera corrente viene unita tramite or logico quella del bianco. Vale la stessa cosa per il giallo con il nero;
3) Per ogni colore principale dei coni (rosso, blu, giallo), sono state applicate le seguenti operazioni una alla volta in sequenza:
    a. Gaussian Blur per eliminare il grosso del rumore. Per il giallo, vista la ridotta dimensione dei coni, tale filtro è stato alleggerito;
    b. Dilatazione per unire aree vicine, ottenendo sagome complete;
    c. Riempimento dei buchi all’interno di un’area;
    d. Smussamento degli angoli;

Vengono successivamente salvati i contorni così trovati dentro un’apposita struttura dati e filtrati ulteriormente per proprietà geometriche. I filtri geometrici applicati si dividono in due parti:

1) Controllo che i bordi approssimati di ogni contorno siano compresi nell’intervallo $[3,9]$. La ragione del limite superiore può dipendere dal fatto che i coni hanno una base quadrata, che aggiungendosi ai lati del cono effettivo arrivano, contando eventuali errori di approssimazione, a 9;
2) Superato il primo controllo vengono calcolati l’area e l’aspect ratio $\left(\frac{box.width()}{box.height()}\right)$ di ogni contorno. Vengono esclusi come potenziali coni i contorni le cui aree sono più piccole di un valore dinamico aggiornato a seconda del colore corrente che viene esaminato (vedi codice) e con un aspect ratio inferiore a 1. Questa assunzione prevede che tutti i coni siano in piedi nel tracciato;

Un problema riscontrato nell’applicazione di questa pipeline deriva dal fatto che, avendo unito due colori diversi, inizialmente delle aree separate che rientrano nello spettro del nero o bianco dell’immagine e rispettano le proprietà geometriche, venivano rilevate come coni, anche se di fatto si trattava di pezzi di cielo, edifici oppure erba circostante il tracciato. Per risolvere questo problema, dopo i filtri geometrici, è stato applicato un ulteriore filtro che controlli, dall’immagine originale, le percentuali del colore principale (1) e quello secondario (bianco o nero) presenti nel contorno. Se le soglie sono rispettivamente maggiore e inferiore di 0.1, allora si tratta di un cono. Questa pipeline è stata ulteriormente rafforzata con tuning sulle soglie dei colori tramite uno script che rileva il colore puntato dal mouse sull’immagine, mentre i kernel utilizzati per le operazioni di cv::*dilate*, cv::*morphologyEx* (open e close) sono tutti di dimensione rispettivamente 5x5, 5x5 e 3x3. Fa eccezione il colore blu, dove per l’operazione di cv::*morphologyEx(open)* è stato utilizzato un kernel 9x9 per rendere più compatti i profili dei coni blu più distanti. I coni gialli e blu più distanti dalla macchina non sono stati rilevati, poiché il loro profilo diventa così piccolo da essere indistinguibile da una macchia di prato oppure un pezzo di cielo. Tale problema potrebbe essere risolto utilizzando una CNN addestrata con molti frames scattati da diverse angolazioni di più varianti dello stesso percorso.

Figura 1.0: notare che i coni vengono evidenziati con una box diversa in base alla loro classificazione

LIVELLO 3: CONE CLASSIFICATION

I coni precedentemente trovati sono stati classificati in base al loro colore principale (1). Poiché le maschere dei vari colori vengono processate una alla volta, quando viene rilevato un cono, è stato sufficiente controllare il colore corrente per riuscire a stabilire le soglie dell'ultimo filtro e classificarlo di conseguenza in caso positivo. L'implementazione di questo livello ha migliorato ulteriormente quello precedente, permettendo di prevenire elementi come le spie del volante, chiazze d'erba e pezzi di cielo uniformi come coni.

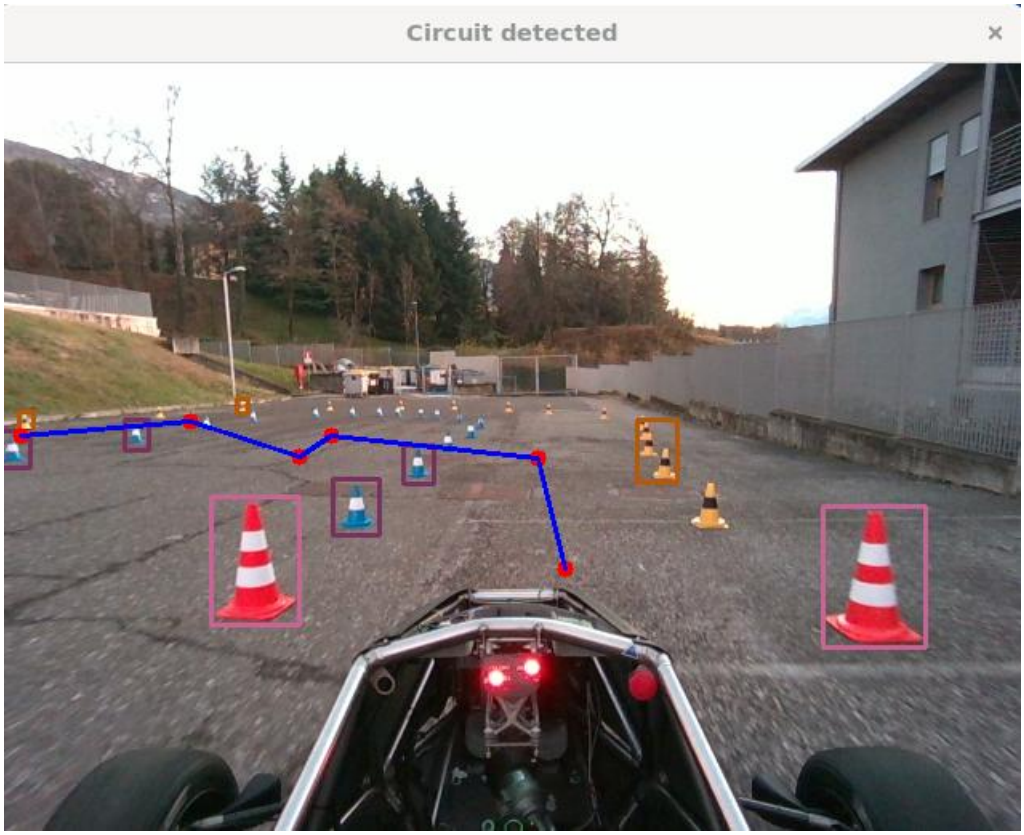
LIVELLO 4: EXTRACTION OF TRACK EDGES

In questo livello sono stati utilizzati i baricentri dei contorni trovati nel livello 2 per determinare l'andamento del tracciato. Ogni volta che viene rilevato un nuovo cono, dopo averlo evidenziato nell'immagine originale, è stato salvato il suo baricentro in un vettore contente i coni di uno stesso colore. L'idea di base alla base dell'implementazione è la seguente:

- Si calcola il punto intermedio tra il baricentro dei due coni rossi. Verrà chiamato starti_point per comodità;
- Per ogni cono blu di baricentro (a, b):
  - Per ogni cono giallo di baricentro (c, d):
    - Si calcola la distanza euclidea tra (a, b) e (b, c);
    - Si considera la distanza minore trovata finora;
  - Si calcola il punto intermedio in questa distanza;

    - Si salva il punto precedentemente trovato su un vettore contenente i punti del tracciato chiamato circuit_points;
- Ripetere il punto precedente un'altra volta invertendo coni blu e gialli per avere più punti.

Sapendo che il circuito parte dai coni rossi, si ordinano i punti di circuit_points per distanza euclidea minore rispetto allo starting_point. Nell'immagine, si disegnano i punti e li si unisce con una linea retta a coppie di due partendo da starting_point. Poiché i coni non vengono rilevati nell'ordine che essi occupano all'interno del circuito, la logica di ordinamento proposta permette di ottenere il profilo approssimato più fedele al tracciato. Come già spiegato nei dettagli del livello 2, non tutti i coni più distanti vengono rilevati. Questo comporta un'approssimazione che può andare fuori dalle linee del tracciato. Nonostante ciò, i coni più vicini vengono rilevati correttamente. Segue che, se si aumentasse la frequenza dei frames, sarebbe possibile ottenere un'approssimazione fedele dei bordi del circuito sulle vicinanze, garantendo che la macchina non vada fuori strada. Per migliorare questo punto è sufficiente perfezionare il livello 2. Successivamente, avendo la possibilità di rilevare più coni, si potrebbe controllare che, ogni volta si sta per disegnare una linea, che essa non intersechi l'allineamento di due o più coni blu e gialli.



LIVELLO 5: ODOMETRY

Il livello 5 della task di recruiting assegnata prevede di calcolare lo spostamento effettuato dalla macchina tra i due frame forniti. Per fare ciò è stato utilizzato orb con parametro 2000, al fine di ottenere più informazioni possibili. Successivamente sono stati estratti i punti chiave e i descrittori dalle due immagini e abbinati utilizzando un Brute Force matcher. Dopo aver filtrato i risultati in base alla loro distanza, i punti chiave sono stati resi 2D. È stata successivamente calcolata la matrice di posa usando quella essenziale fornita dall'esercizio, dalla quale sono state ricavati la rotazione e traslazione, le quali sono poi state stampate a video. In questo punto non sono state riscontrate particolari difficoltà. Si potrebbe migliorare questo livello aumentano l'Informatività del risultato, leggendo ogni valore e confrontandolo col precedente al fine di dedurre la direzione del movimento.

```
-----------------------------------
   Runtime Environment Information
-----------------------------------
OpenCV version : 4.5.4
Compiler       : GCC 11.4.0
C++ standard   : C++17
-----------------------------------

Rotation:
[0.9997229413479667, -0.004477265604383662, 0.02310832393917968;
 0.004482730043727903, 0.9999899354626655, -0.0001846746574483144;
 -0.02310726452709897, 0.0002882118697189803, 0.9997329499721377]
Traslation:
[-0.9206766868554274;
 -0.1255971506383617;
 -0.3695670359115353]
```

PRECISAZIONI:

1) Colori principali dei coni: rosso, blu e giallo;